Mittuniversitetet
MID SWEDEN UNIVERSITY

Estimation and Post-Capture Compensation of Synchronization Error in Unsynchronized Multi-Camera Systems

Mid Sweden University Technical Report

Elijs Dima, Yuan Gao, Mårten Sjöström,
Roger Olsson, Reinhard Koch, and Sandro Esquivel

E. Dima, M. Sjöström and R. Olsson are with the Inst. of Information Systems and Technologies, Mid Sweden University, 85170 Sundsvall, Sweden (Email:[name.lastname]@miun.se).

Y. Gao, R. Koch and S. Esquivel were with the Institute of Computer Science, Christian-Albrechts-University of Kiel, 24118 Kiel, Germany at the time of research and writing (Email: [yga,rk,sae]@informatik.uni-kiel.de).

Project research and drafting period: 2017 - 2018.

Project funding:
- Grant 6006-214-290174 from Rådet för Utbildning på Forskarnivå (FUR), Mid Sweden University
- LIFE project grant 20140200 of the Knowledge Foundation, Sweden
- European Union's Framework Programme for Research and Innovation Horizon 2020 (2014-2020) under the Marie Skłodowska-Curie Actions Grant Agreement No. 676401

Abstract

Multi-camera systems are used in entertainment production, computer vision, industry and surveillance. The benefit of using multi-camera systems is the ability to recover the 3D structure, or depth, of the recorded scene. However, various types of cameras, including depth cameras, can not be reliably synchronized during recording, which leads to errors in depth estimation and scene rendering. The aim of this work is to propose a method for compensating synchronization errors in already recorded sequences, without changing the format of the recorded sequences. We describe a depth uncertainty model for parametrizing the impact of synchronization errors in a multi-camera system, and propose a method for synchronization error estimation and compensation. The proposed method is based on interpolating an image at a desired timeframe based on adjacent non-synchronized images in a single camera's sequence, using an array of per-pixel distortion vectors. This array is generated by using the difference between adjacent images to locate and segment the recorded moving objects, and does not require any object texture or distinguishing features beyond the observed difference in adjacent images. The proposed compensation method is compared with optical-flow based interpolation and sparse correspondence based morphing, and the proposed synchronization error estimation is compared with a state-of-the-art video alignment method. The proposed method shows better synchronization error estimation accuracy and compensation ability, especially in cases of low-texture, low-feature images. The effect of using data with synchronization errors is also demonstrated, as is the improvement gained by using compensated data. The compensation of synchronization errors is useful in scenarios where the recorded data is expected to be used by other processes that expect a sub-frame synchronization accuracy, such as depth-image-based rendering.

Index Terms

Multi-camera systems, Synchronization, Multiview, 3D Acquisition, Video alignment, Depth uncertainty

I Introduction

Since the rise of Three-Dimensional (3D) cinema and television in the 20th century, systems of multiple cameras have been used for recording in order to preserve the scene depth. Nowadays, multi-camera recording persists in the domains of computer vision, robot vision, entertainment production, and surveillance. Computer vision applications, such as behavior observation [1] and motion capture [2], use multiple cameras to eliminate self-occlusions. In robotics and robot vision, multi-camera systems are used for 3D mapping [3] and mobile robot self-positioning [4], [5]. In entertainment production, depth and 3D information is used to produce cinematic effects [6], [7] and Multi-View (MV) video [8], [9]. In surveillance, methods exist for merging multi-perspective video streams in order to improve coverage and detail retention [10] or reconstruct virtual replicas of real environments [11]. Multi-camera systems are diverse, and contain professional video recorders [8], specialty depth cameras [12]–[14], small low-cost cameras [5], and surveillance cameras [15]. Some of these camera types do not support a synchronization control signal. Synchronization in multi-camera systems is necessary so that all cameras sample the scene's

light field at the same time. Incorrect synchronization leads to recorded data inconsistency, which in turn may lead to errors in depth estimation, scene reconstruction and rendering.

Camera synchronization directly affects the result of all operations that integrate multiple camera video streams. Cameras with hardware inputs for a synchronization signal tend to be more expensive than non-synchronized (free-running) alternatives, or result in larger system bandwidth or lower temporal sampling rate [16]. Moreover, depth cameras, such as the low-cost Structured Light (SL) and Time-of-Flight (ToF) Kinect sensors [17], do not support hardware synchronization. At the same time, applications such as Depth-Image Based Rendering (DIBR) expect all input data to be recorded by synchronized camera systems. Thus, there is a gap between non-synchronized camera systems and applications that use output data from multi-camera systems. When sensors in a multi-camera system cannot be synchronized, there is a need for post-capture data synchronization. Since these processes come at a financial or computational cost, there is also a need to determine *a priori* whether a multi-camera system requires additional synchronization efforts for given requirements.

The research questions for this study are as follows: **1**: How can one determine, prior to capturing specific datasets, what consequences will arise from using cameras with an assumed level of synchronization error? **2**: How can synchronization errors between depth and color cameras be estimated and used to improve the recorded color or depth content, such that conventional depth reconstruction & rendering methods can benefit?

In this paper we present two new contributions. First, we present new research on the depth uncertainty estimation of multi-camera systems, based on the model proposed in [18]. We examine how the depth uncertainty model can relate synchronization error to rendered image quality, thereby addressing research question **1**. This contribution covers new assessments of the depth uncertainty model and extends it to describe maximum allowable synchronization error under given quality requirements. Second, we describe a novel post-capture synchronization method ("re-synchronization"), which consists of a correspondence-based sequence alignment and a weighted frame interpolation for Multi-View plus Depth (MVD) sequences, thereby addressing research question **2**. The proposed method produces synchronized sequences, where frame content from all involved cameras represents the same point in time and therefore, the same scene geometry. In particular, the method is aimed at sub-frame synchronization error compensation in depth data to match recorded color sequences, without affecting the depth data format.

Outline: Section II mentions the related work. Section III describes Depth Uncertainty estimation, synchronization error estimation, and synchronization error compensation in MVD sequences. We explain the experimental setup in Section IV. The experiment results are shown and discussed in Section V. Section VI concludes the work by discussing the result implications and future work.

II Related Work

The previous section introduced the research problem and the focus of our work on camera synchronization. This section covers the related work on synchronization in multi-camera systems, touching on the need for synchronization, and the ways in which synchronization errors are addressed.

The need for synchronization: In the literature, there are three general ways of approaching the question of synchronization necessity. To the best of our knowledge, the question of whether synchronization is strictly necessary for a given camera setup has not yet been focused upon in related literature, except for [18]. Instead, the need for synchronization is typically assumed to be true (as seen in most papers cited after this paragraph), abstracted into a wider questioning of sensor and depth data enhancement [12], or excluded from discussion as not part of a multi-camera system pipeline (e.g. in surveys such as [19], [20]). The model described in [18] relates synchronization error to a potential error in estimating 3D positions of in-scene objects, when using triangulation from multiple Two-Dimensional (2D) camera views. The model allows to find out how precise the synchronization between cameras needs to be, given an expected 3D accuracy.

Addressing synchronization errors: Within the literature subset that *does* address synchronization, there are three approaches on *how* to use desynchronized cameras in a multi-camera system: 1) synchronization error is eliminated at the source, i.e. some level of synchronization is ensured between cameras. 2) The post-capture re-synchronization or *alignment* of video sequences is treated as a distinct, solvable problem. 3) Re-synchronization directly into higher-level applications (e.g. calibration, camera tracking, image rendering).

(1) The first approach of solving synchronization problems is to prevent them. During recording, synchronization can be achieved by using special-purpose hardware [8], [21]. If the camera hardware does not support a synchronization signal input, then synchronization errors have two sources. An *unsynchronized "record image" instruction* to several cameras can cause an initial synchronization offset, and *camera internal clock drift* can cause an increasing synchronization error in long-running sequences [22]. Camera clock drift can be solved by replacing long continuous recordings with shorter, sequential recordings [23]. The remaining problem of initial synchronization offsets is then solved by timestamp propagation through a network, followed by video stream buffering [24], frame dropping [25], or staggered stream re-launching [23], [26].

(2) The second approach is to focus on sequence realignment. Among works dealing with video sequence realignment as a distinct problem, a minority of approaches try to align video sequences recorded at different times (e.g. different days) [27]. However, such approaches step outside the domain of a multi-camera system problem. In a multi-camera system constraint, video sequence alignment methods rely on either recorded meta-data, or natural in-scene information to estimate the synchronization error. Methods using metadata for sequence alignment tend to add additional recorded dimensions such as audio signals [28], or rely on encoded video bitrate-per-frame correlation [29]. Some content-based

methods rely on intensity difference minimization [30], [31] or add deliberate timing cues such as camera flashes in the scene [32], however most content-based methods rely on nondescript feature identification [33]–[39].

The tri-focal tensor approach by Lei et al. [33] uses a Levenberg-Marquardt (LM) optimization on a tri-focal tensor on three frames, obtained from point and line correspondences in three video sequences. The optimization steps are performed on frames and globally on the image sequences, thereby providing a synchronization offset estimate. The estimated offset's accuracy is below the time interval between two frames (called "sub-frame accuracy"). However, the frames within the sequences are not altered in any way - the output of the method is just the set of synchronization offsets.

A nearest-frame video sequence alignment proposed in [34] uses a strict relation between back-projected lines of five independent points matched in two sequences. Expressing the fifth point as a linear combination of the other four points produces a distinctive per-frame position. Across the sequences, the fifth point's position produces a virtual moving trajectory, which is used as basis for sequence alignment (i.e. offset estimation).

In [35], the synchronization problem is redefined as a problem of spatial alignment for sequences containing similar planar motions. The object motions are computed through feature detection in each sequence. The spatial correspondence between two videos is obtained using an affine transform and a probabilistic model. The temporal alignment is then computed based on a Dynamic Time Warping technique, using the probabilistic value as input factor. The method's result is an estimated synchronization offset for each sequence.

Evangelidis et al. [36] also focus on video sequence alignment. A reference sequence is indexed through a geometric hash function, which encodes and describes neighbouring feature points in a video frame. Sequence alignment is performed as a best-match query of the geometric indexing between the reference and target sequences.

Information on epipolar lines is used to align sequences in [37]. Intensity and color based temporal signals on epipolar lines are matched using a probabilistic optimization framework. Although there is no direct correspondence search, determining the epipolar lines *a priori* requires finding the fundamental calibration matrix between two cameras, which in turn usually requires finding static matching features in the sequences. The proposed method produces temporal alignment information for video sequences, and does not adjust the sequence contents.

Diego et al. [30] use a Bayesian network to treat sequence alignment as a probabilistic labelling problem, i.e. a "maximum *a posteriori*" inference problem. Given two independently moving cameras, both image intensity and camera position/trajectory data (in the described case, given by a Global Positioning System tracker) are used to constrain the possible sequence alignment. Intensity-based sequence alignment is also used in [31], where spatio-temporal alignment between two sequences is iteratively improved through a pyramid-based search window scaling approach; where each step minimizes the intensity difference between the overlapping sections of both sequences.

(3) Lastly, instead of explicit sequence alignment, some works solve synchronization indirectly, through addressing a different main problem. Such problems are camera location and movement estimation [4], depth-guided correspondence estimation [40], or two-camera geometry estimation [22], [41], [42].

Kerl et al. [4] use a joint camera with unsynchronized color and depth sensors to reconstruct the trajectory of the camera movement in a static space. Rolling shutter artifacts are compensated by using a scanline offset based on known timestamps, and camera trajectory is constantly updated based on B-splines through dense image alignment. The compensation of the time-shift (or synchronization offset) between depth and color data is a by-product of the B-spline-guided depthmap warping to color images.

Ruhl et al. in [40] propose a depth-guided correspondence estimation method that uses both low-resolution depthmaps, user input [43] and geometric proxies to guide a dedicated high-resolution image correspondence estimation. This method relies on a coarse-to-fine image pyramid to resolve correspondence match uncertainties, and uses the resulting dense image-to-image correspondence maps as a main contributor to a rendering process. Unsynchronized depthmaps from a ToF sensor are treated as an approximate reference to be refined by the dense correspondence maps. Thus, the influence of temporally offset depth data is reduced. A rendering toolchain has also been proposed in [44], but no reference implementations have been made available.

Albl et al. [22] (with additional method information in [45]) propose a new two-view geometry estimation method that includes approximated time-shift, similar to Noguchi et al. [41] and Nischt et al. [42]. All three methods rely on using proposed image point trajectories to solve the fundamental matrix. However, Albl et al. show a minimal solution to epipolar geometry between two cameras, that requires only 5 samples of a moving point to create a homography, and 8 samples to recover the fundamental matrix through a system of 18 polynomial equations in 6 unknown parameters. This allows using Gröbner basis solvers [46] for solving the constrained polynomial equation system, and Random Sample Consensus (RANSAC) [47] for filtering out estimated point trajectory samples that do not follow a straight path at linear velocity. While the method of Albl et al. can be used explicitly for synchronization, the overall proposal in [22] aims at camera geometry determination, with integrated synchronization solving.

Summary: One set of methods manipulate the capture process at capture time, in order to ensure that the recorded data is synchronous. Another set of methods takes non-synchronized data and estimates the temporal offset, as information to be used in a later, unspecified data consumer process. A third set of methods creates processes (calibration, rendering, positioning) that implicitly compensate for the temporal offsets evident in data. The notable omissions are two-fold. First, there is a lack of methods that motivate whether synchronization is necessary in a given scenario. Second, few synchronization methods seek to compensate (or correct) the unsynchronized data on a subframe level, without immediately consuming the data in the process - especially in the context of MVD data.

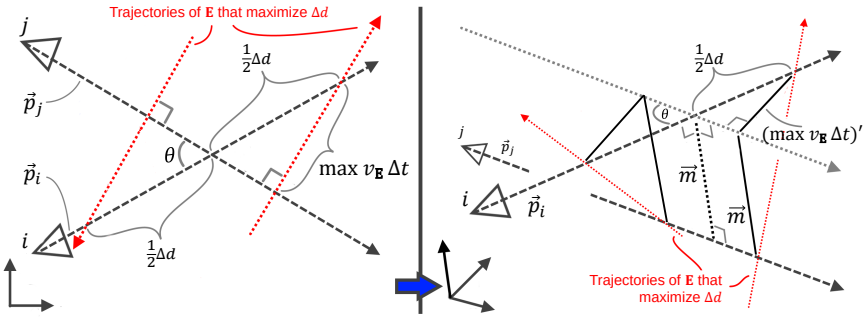


Fig. 1. Geometric basis for deriving depth uncertainty Δd. Rays $\overrightarrow{p}_i$, $\overrightarrow{p}_j$ have recorded a moving point **E**. Shown trajectories of **E** (red) maximize the depth uncertainty along the ray $\overrightarrow{p}_i$. Left: rays $\overrightarrow{p}_i$, $\overrightarrow{p}_j$ are intersecting. Right: $\overrightarrow{p}_i$, $\overrightarrow{p}_j$ are not intersecting.

III Proposed Method

The previous section covered the related work on synchronization, and highlighted the need for knowing how much synchronization error can be tolerated in a given camera system, and for methods that explicitly correct non-synchronized multi-view data. In this section, we cover depth uncertainty (based on [18]), and propose a method for synchronization error estimation and synchronization error compensation.

III-A Depth Uncertainty

In a MV scenario, the 3D position of a scene point ($\mathbf{E} = [\mathbf{E}_1, \mathbf{E}_2, \mathbf{E}_3]^{\mathrm{T}}$) is determined by multi-view geometry, which is based on triangulation. A single 2D image records the transverse positions of **E**, and places no constraints on the depth between **E** and the camera. A second 2D image is used to triangulate **E**'s depth from the first camera. This triangulation can be inaccurate, if the cameras are not synchronized and **E** is moving. Because of the synchronization error between cameras, the depth of the moving point **E** remains uncertain. However, knowing the synchronization error allows to set minimum and maximum constraints on **E**'s possible depth.

Depth uncertainty (Δd) is the difference between the minimum and maximum possible depths between the observed point and the observing camera's center [18]. Figure 1 shows two cameras i and j, that observe a moving point **E** along rays $\overrightarrow{p}_i$ and $\overrightarrow{p}_j$, respectively. If the cameras are synchronized, **E** is at the intersection of $\overrightarrow{p}_i$ and $\overrightarrow{p}_j$. If the cameras are not synchronized, **E** can be at any point on the ray $\overrightarrow{p}_i$ near the intersection, within the interval of size Δd as show in Fig. 1, left.

The depth uncertainty can be calculated by:

$$\Delta d = \begin{cases} \dfrac{2\sqrt{\left(\max v_{\vec{E}}\Delta t\right)^2 - \|\vec{m}\|^2}}{\sin(\theta)} & \text{, if } (\max v_{\vec{E}}\Delta t)^2 \geq \|\vec{m}\|^2, \\ \text{-undefined-} & \text{otherwise,} \end{cases} \tag{1}$$

where θ is the angle between rays. In a 3D case where the rays may not intersect, $\vec{m}$ is the shortest path between $\overrightarrow{p}_i$ and $\overrightarrow{p}_j$. The maximum speed that **E** can have is $\max v_{\vec{E}}$, and Δt is the synchronization error between i and j. This means that $\max v_{\vec{E}} \Delta t$ is the longest distance in 3D space that **E** can cover between being recorded by i and j. The "undefined" case in Equation (1) occurs when the path $\vec{m}$ is larger than the longest path that **E** can travel.

The angle θ between rays $\overrightarrow{p}_i$ and $\overrightarrow{p}_j$ can be determined from the inner product of their vectors. We note that $\vec{p}_i$ represents the normalized vector of the ray $\overrightarrow{p}_i$, which gives:

$$\theta = \arccos\left(\frac{\vec{p}_i \cdot \vec{p}_j}{\|\vec{p}_i\| \ \|\vec{p}_j\|}\right) . \tag{2}$$

The vectors $\vec{p}_i, \vec{p}_j$ are found by using the pinhole camera model [48], which describes camera sensor and lens parameters in an intrinsic matrix **K**, camera position in 3D space in vector $\vec{C}$, and rotation in matrix **R**. The rays $\overrightarrow{p}_i$, $\overrightarrow{p}_j$ are:

$$\overrightarrow{p}_i = \vec{C}_i + \lambda \mathbf{R}_i^{-1} \mathbf{K}_i^{-1} c_i \ , \tag{3}$$

where $c_i = (x, y, 1)^{\mathrm{T}}$ is the image coordinate where the ray $\overrightarrow{p}_i$ intersects the sensor of camera i. Ray $\overrightarrow{p}_j$ is defined by the same method. Setting the scale factor $\lambda = \{0, 1\}$ in Equation (3) for start and end points gives the normalized vectors $\vec{p}_i, \vec{p}_j$:

$$\vec{p}_i = \mathbf{R}_i^{-1} \mathbf{K}_i^{-1} c_i \ , \ \vec{p}_j = \mathbf{R}_j^{-1} \mathbf{K}_j^{-1} c_j \ . \tag{4}$$

The path $\vec{m}$ is found by defining a vector $\vec{p}_o$, which goes from the origin of ray $\overrightarrow{p}_j$ to the origin of ray $\overrightarrow{p}_i$. Applying the method described by Lumelsky [49] gives:

$$\vec{m} = \vec{p}_o + \frac{(\hat{b}\hat{e} - \hat{c}\hat{d})\vec{p}_i - (\hat{a}\hat{e} - \hat{b}\hat{d})\vec{p}_j}{\hat{a}\hat{c} - \hat{b}^2}, \tag{5}$$

$$\text{where } \hat{a} = \vec{p}_i \cdot \vec{p}_i, \ \hat{b} = \vec{p}_i \cdot \vec{p}_j, \ \hat{c} = \vec{p}_j \cdot \vec{p}_j, \ \hat{d} = \vec{p}_i \cdot \vec{p}_o, \ \hat{e} = \vec{p}_j \cdot \vec{p}_o \ .$$

Equation (1) fits a context where $\max v_{\mathbf{E}}$ and Δt are inputs, given by the *a priori* knowledge about the camera system and its intended application. Eq. (1) describes a discrete instance of **E** in a discrete dataset. To parametrize a multi-camera system's general depth uncertainty, Equation (1) is assessed for all possible combinations of rays. The general depth uncertainty can then be calculated as the mean $\overline{\Delta d}$:

$$\overline{\Delta d} = \frac{1}{n} \sum_{k=1}^{n} \Delta d_k \ , \text{ where } \Delta d_k \in \{\Delta d \mid \forall \ (\overrightarrow{p_i}, \overrightarrow{p_j} \implies \Delta d) \ \} \ . \tag{6}$$

The notation $(\overrightarrow{p_i}, \overrightarrow{p_j} \implies \Delta d)$ means that the rays $\overrightarrow{p_i}, \overrightarrow{p_j}$ result in a defined Δd, according to Equation (1).

III-B Synchronization Error Estimation

In Section III-A, Δt was used as a general parameter. In order to compensate synchronization errors between two cameras (i, j), we now consider Δt explicitly. The synchronization error Δt_n is:

$$\Delta t_n = \| t_n^i - t_n^j \| \ , \tag{7}$$

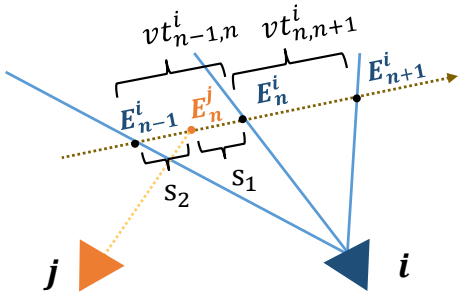


Fig. 2. A moving point **E** observed by depth camera j at frame n, and by a color camera i at frames $n-1, n, n+1$.

where t^i_n and t^j_n are the times when the n-th frame is recorded by cameras i and j, respectively.

If cameras i, j are not synchronized, they record the moving point **E** at different real positions ($\mathbf{E}^i_n$ and $\mathbf{E}^j_n$) in the n-th frame. The difference (s_1 in Fig. 2) between these positions is given by:

(8)

This equation relies on the assumption that **E**'s trajectory is sufficiently linear over short timescales. The same assumption underpins several successful video sequence alignment methods surveyed in Section II, such as [34], [37], [45]. Assuming that multiple frames record **E** in sequence, Equation (8) gives a way to estimate Δt as a ratio of positions:

$$\frac{s_1}{s_1+s_2} = \frac{v\Delta t_n}{v\|t^i_n - t^i_{n-1}\|} = \frac{\Delta t_n}{1/\psi(i)} \implies \Delta t_n = \frac{s_1}{(s_1+s_2)\psi(i)}\ , \tag{9}$$

In order to use Equation (9), one must know the 3D positions of $\mathbf{E}^i_n$, $\mathbf{E}^j_n$, and $\mathbf{E}^i_{n-1}$. If camera j is a depth camera, then $\mathbf{E}^j_n$ is already known. To find $\mathbf{E}^i_n$ and $\mathbf{E}^i_{n-1}$, we only need to find a line of best fit (ω) that intersects $\mathbf{E}^j_n$ and satisfies the following constraint:

$$\frac{\|\mathbf{E}^i_{n-1} - \mathbf{E}^i_n\|}{\|\mathbf{E}^i_n - \mathbf{E}^i_{n+1}\|} = \frac{\|t^i_{n-1} - t^i_n\|}{\|t^i_n - t^i_{n+1}\|}\ . \tag{10}$$

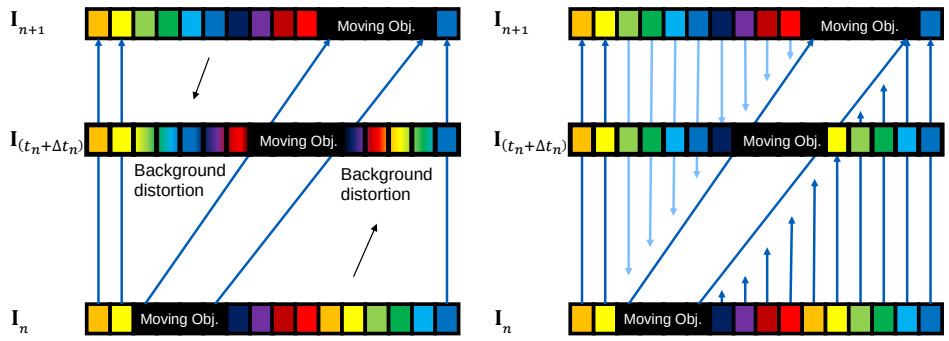
If camera j is not a depth-sensor, then $\mathbf{E}^i_n$, $\mathbf{E}^j_n$, and $\mathbf{E}^i_{n-1}$ can be found if they belong to a planar surface $\vec{\Omega} = [\Omega_1, \Omega_2, \Omega_3, \Omega_4]^{\mathrm{T}}$, i.e. any 3D point $\mathbf{E} = [X, Y, Z]^{\mathrm{T}}$ on this plane should meet:

$$\begin{bmatrix}\mathbf{E}^{\mathrm{T}} & 1\end{bmatrix}\vec{\Omega} = 0\ . \tag{11}$$

Applying the pinhole camera model [48] to Equation (11) gives:

$$\begin{bmatrix} f_x & 0 & c_x - u \\ 0 & f_y & c_y - v \\ \Omega_1 & \Omega_2 & \Omega_3 \end{bmatrix}\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -\Omega_4 \end{bmatrix} \implies \mathbf{E} = \left(\mathbf{K} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ \Omega_1 & \Omega_2 & \Omega_3 \end{bmatrix} + \boldsymbol{e}\begin{bmatrix} 0 & 0 & -1 \end{bmatrix}\right)^{-1}\begin{bmatrix} 0 \\ 0 \\ -\Omega_4 \end{bmatrix} \tag{12}$$

If **E** is uniformly moving on the $\vec{\Omega}$ plane, Equation (12) can be applied for transforming a 2D tracking point e^j on camera j image plane to the 3D point $\mathbf{E}^j$ on the $\vec{\Omega}$ plane in the camera coordinate system of



(a) Morphing (b) Proposed Method

Fig. 3. Interpolation of intermediate image $\mathbf{I}_{(\mathbf{t_n}+\mathbf{\Delta t_n})}$, with moving foreground object recorded in images $\mathbf{I}_n$ and $\mathbf{I}_{n+1}$. Background distortion in morphing compared to background recovery in the method proposed in Equation (14)

camera j. In practice, the line of best fit (ω) for points $\mathbf{E}_n^i$, $\mathbf{E}_n^j$, and $\mathbf{E}_{n-1}^i$ can be estimated using the least squares method.

III-C Synchronization Error Compensation

Once the synchronization error Δt between cameras i and j is known, the primary task is to compensate the synchronization error in the captured images on one of the cameras. For convenience, we omit specifying i or j, and assume camera j unless stated otherwise. Compensating the synchronization error in an image means to create a target image $\mathbf{I}_{(t_n+\Delta t)}$ from recorded images $(\mathbf{I}_n, \mathbf{I}_{n-1}, \ldots)$. Therefore, compensating Δt the n-th frame can be approximated by:

$$\mathbf{I}_{(t_n+\Delta t)} \simeq \mathbf{I}_n + \delta_n \mathbf{V}_{n,n+1}, \tag{13}$$

which requires a factor δ_n to describe the temporal ratio between the images, and a tensor $\mathbf{V}_{n,n+1}$ describing how to change pixels from n-th to $n+1$-th frame. Figure 3 (left) shows what happens if Equation (13) is implemented as mesh-based or field-based morphing [50]–[52]: around the edges of moved geometry, the background is deformed. To avoid this, we propose a tensor interpolation on each pixel $(x, y)^\mathsf{T}$ of the target image $\mathbf{I}_{(t_n+\Delta t)}$:

$$\mathbf{I}_{(t_n+\Delta t)}(x,y) = \begin{cases} \mathbf{I}_n(x',y') + \delta_n \mathbf{V}_{n,n+1}(x',y',3), & \text{if } \triangle \\ \mathbf{I}_{n+1}(x,y) & \text{otherwise;} \end{cases}$$

$$\triangle\colon \exists \begin{pmatrix} x' \\ y' \end{pmatrix} \text{ s.t. } \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x' + \delta_n \mathbf{V}_{n,n+1}(x',y',1) \\ y' + \delta_n \mathbf{V}_{n,n+1}(x',y',2) \end{pmatrix}. \tag{14}$$

The tensor $\mathbf{V}_{n,n+1}$ is equivalent to a matrix of per-pixel vectors, where each vector contains $\Delta x, \Delta y$ and Δz:

$$\mathbf{V}_{n,n+1}(x,y) = [\Delta x, \Delta y, \Delta z]\ . \tag{15}$$

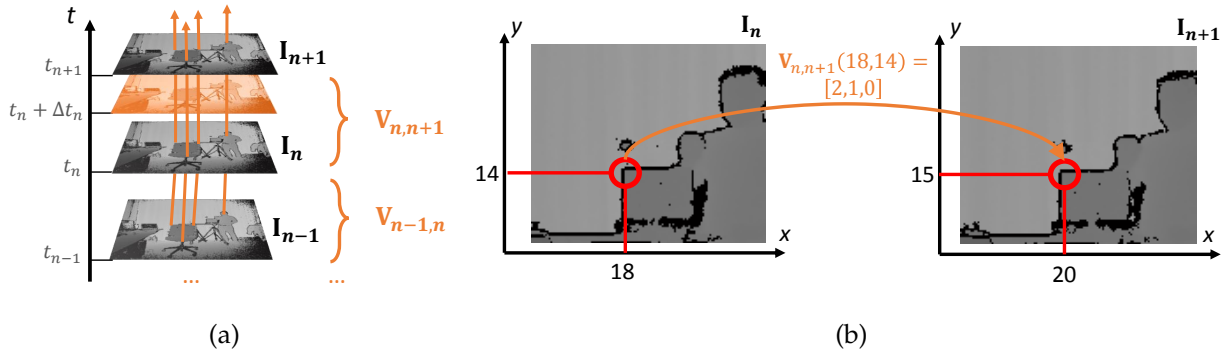


Fig. 4. a) Temporal depth adjustment to desired time $t_n + \Delta t_n$ with images $\mathbf{I}_n, \mathbf{I}_{n+1}$ captured at times t_n, t_{n+1}, using the vector tensor $\mathbf{V}_{n,n+1}$. b) Example of constructing the tensor $\mathbf{V}_{n,n+1}(18,14) = [\Delta x, \Delta y, \Delta z]$ from $\mathbf{I}_n$ and $\mathbf{I}_{n+1}$.

The two variables $\Delta x, \Delta y$ record the position change in 2D image space (where x, y are standard image space axes, or pixel row & column indices), and Δz records the change in pixel value. Figure 4 shows an example of such a vector between two images. Notation "$\mathbf{V}_{n,n+1}(x,y,3)$" means the 3-rd element (Δz) from the vector $\mathbf{V}_{n,n+1}(x,y)$.

The scaling factor δ_n represents where $t_n + \Delta t_n$ is between t_n and t_{n+1}. This can be determined by:

$$\delta_n = \frac{\Delta t_n}{\Delta t_n + \frac{1}{\psi(i)} - \Delta t_{n+1}} \ , \tag{16}$$

where $\psi(i)$ is the framerate of camera i. If camera j's framerate $\psi(j)$ is a known constant, then Equation (16) does not need to reference camera i, and reduces to:

$$\delta_n = \frac{\Delta t_n}{1/\psi(j)} \ . \tag{17}$$

The tensor $\mathbf{V}_{n,n+1}$ can be estimated from frames $\mathbf{I}_n$ and $\mathbf{I}_{n-1}(,\ldots)$, if $\mathbf{I}_{n+1}$ is not available. Assuming that the recorded moving objects have inertia, the scene changes in time interval $[t_n, t_{n+1}]$ are likely related to the scene changes in intervals $[t_{n-1}, t_n]$, $[t_{n-2}, t_{n-1}]$, ... Thus, the tensor can obtained by:

$$\mathbf{V}_{n,n+1} = -\left(\kappa_1 \frac{\mathbf{V}_{n,n-1}}{1} + \ldots + \kappa_k \frac{\mathbf{V}_{n,n-k}}{k}\right) \ , \text{ s.t. } \sum_{m=1}^{k} \kappa_m = 1 \ , \tag{18}$$

where κ is a non-zero weight dependent on the scene, and k is the number of images used for $\mathbf{V}_{n,n+1}$ estimation.

If $\mathbf{I}_n$ and $\mathbf{I}_{n+1}$ are 2D color images, then $\mathbf{V}_{n,n+1}$ is the optical flow [53], [54] between $\mathbf{I}_n$ and $\mathbf{I}_{n+1}$. If $\mathbf{I}_n, \mathbf{I}_{n+1}$ are depthmaps, then optical flow algorithms may fail due to lacking unique gradients and surface textures. In such cases, we propose generating the tensor $\mathbf{V}_{n,n+1}$ by using the difference $\mathbf{I}_d$ between $\mathbf{I}_n$ and $\mathbf{I}_{n+1}$ to drive a segmentation of moving objects and per-object pixel mapping. Algorithm 1 shows the proposed approach, consisting of segmentation and pixel-matching steps.

The segmentation function in Algorithm 1 uses the difference $\mathbf{I}_d$ to identify the moving object positions in both depthmaps, as shown in Algorithm 2. The positions are stored as binary masks, and passed to

Algorithm 1 Pseudocode for estimating tensor $\mathbf{V}_{n,n+1}$

Require: $\mathbf{I}_n$, $\mathbf{I}_{n+1}$

$\mathbf{I}_d \leftarrow \mathbf{I}_{n+1} - \mathbf{I}_n$

$Seg_n, Seg_{n+1} \leftarrow$ SEGMENT$(\mathbf{I}_n, \mathbf{I}_{n+1}, \mathbf{I}_d)$

$\mathbf{V}_{n,n+1} \leftarrow$ MATCHPIXELS$(Seg_n, Seg_{n+1}, \mathbf{I}_n, \mathbf{I}_{n+1})$

Algorithm 2 Pseudocode of the Segmentation function

function SEGMENT$(\mathbf{I}_1, \mathbf{I}_2, \mathbf{I}_d)$
  Initialize $Seg_1 = \emptyset, Seg_2 = \emptyset, queue = \emptyset$
  for each pixel position $p \in \mathbf{I}_d$ **do** ▷ Find initial positions of moving foreground object
    if $\mathbf{I}_d(p) < 0$ **then** ▷ foreground pixel
      $queue \leftarrow \{queue, p\}$
    end if
  end for
  while $queue \neq \emptyset$ **do** ▷ Expand the foreground object mask in first depthmap
    $p \leftarrow queue(top)$, $queue \leftarrow queue - queue(top)$
    for each neighbor position p_{tmp} of p **do**
      if $\mathbf{I}_1(p_{tmp})$ is similar to $\mathbf{I}_1(p)$ **then**
        $queue \leftarrow \{queue, p_{tmp}\}$
      end if
    end for
    $Seg_1 \leftarrow \{Seg_1, p\}$
  end while
  for each $p \in \mathbf{I}_{n+1}$ **do** ▷ Set the foreground object mask in second depthmap
    if $(p \in Seg_1$ **and** $\mathbf{I}_d(p) \leq 0)$ **or** $\mathbf{I}_d(p) < 0$ **then**
      $Seg_2 \leftarrow \{Seg_2, p\}$
    end if
  end for
  return Seg_1, Seg_2
end function

the pixel-matching function in Algorithm 3. This function uses the segment masks to find the position change $[\Delta x, \Delta y]$ and the value change Δz for each pixel belonging to the moving objects. Using $\mathbf{I}_d$ ensures that the segmentation itself does not need to be accurate with respect to the actual recorded objects, since the difference between depthmaps also defines the difference between the identified segments. Any static areas that get included in the Algorithm 3 will simply produce $[\Delta x, \Delta y, \Delta z] = [0, 0, 0]$ and thus not affect the depthmap compensation process in Equation (14).

Algorithm 3 Pseudocode of the Pixel-matching function

function MATCHPIXELS($Seg_1, Seg_2, \mathbf{I}_1, \mathbf{I}_2$)
  Initialize $\mathbf{V}_{1,2}$ with all elements $= [0,0,0]$
  $\vec{dir} \leftarrow (Seg_2.center - Seg_1.center)$
  for each scanline $line_1$ **s.t.** $line_1 \in \mathbf{I}_1$ **and** $line_1 \parallel \vec{dir}$ **do**
    for each left edge $p_{l,1}$ **do**
      find nearest right edge $p_{r,1}$
      find nearest $p_{l,2}, p_{r,2} \in \mathbf{I}_2$
      for each pixel p_1**s.t.**$p_1 \geq p_{l,1}$ **and** $p_1 \leq p_{r,1}$ **do**
        find respective p_2 between $p_{l,2}, p_{r,2}$
        $(\Delta x, \Delta y) \leftarrow (p_2 - p_1)$
        $\Delta z \leftarrow \mathbf{I}_2(p_2) - \mathbf{I}_1(p_1)$
        $\mathbf{V}_{1,2}(p_1) \leftarrow [\Delta x, \Delta y, \Delta z]$
      end for
    end for
  end for
  return $\mathbf{V}_{1,2}$
end function

IV Test Arrangement and Evaluation Criteria

To verify the proposed compensation method and show that using compensated (re-synchronized) data can improve depth-based image rendering results compared to unsynchronized or aligned-to-nearest-frame data, the following tests have been performed: **1)** Depth Uncertainty Evaluation of two camera systems, **2)** Synchronization Error Estimation test, based on two real datasets recorded by these systems; **3)** Synchronization Error Compensation test on real and artificially-generated depthmap sequences with synchronization errors.

IV-A Depth Uncertainty Evaluation Test

This experiment checks whether $\overline{\Delta d}$, as described in Section III-A, relates to image quality. Two recorded datasets from two different camera systems are used to provide MVD data with synchronization error in depthmap sequences.

First, to show that $\overline{\Delta d}$ relates to Δt, the camera system calibration data and synchronization error Δt is used in Equation (6) to calculate $\overline{\Delta d}$ for each system. Then, synchronization error Δt is changed step-wise and new $\overline{\Delta d}$ are simulated. Step size is selected as $1/\psi$, in order to produce $\overline{\Delta d}$ values corresponding to such synchronization errors, for which a depthmap is available in the dataset.

Second, to demonstrate whether $\overline{\Delta d}$ correlates with a camera system's performance, we show the relation between Δt and depth-based view reprojection quality. The reprojected views are generated thus: a color image $\mathbf{I}_{t_n}$ from camera 1 is selected, together with depthmaps $\mathbf{I}_{t_n+\Delta t}$ for all stepwise-evaluated Δt. For each Δt corresponding to each $\overline{\Delta d}$, the depthmap and the color image is used to project a novel view in camera 2's position, using a conventional DIBR method [55], in two variants - one without any disocclusion in-painting, and one with a naive nearest-background-neighbour pixel in-painting algorithm. The reprojected views at camera 2 position are compared to an actual view $\mathbf{I}_{t_n}$ taken by camera 2, using Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index (SSIM), and Mean Squared Error (MSE).

IV-B Synchronization Offset Estimation Test

In this test, we compare the synchronization error estimation method in Section III-B against a state-of-the-art synchronization offset estimation method by Albl et al. [45], on two real MVD datasets **A** and **B**. Dataset **A** contains the ground truth for the time of capture, in the form of an in-scene clock. For depthmaps, we assume that the Kinect infrared and RGB sensors have matched exposure start and stop times. The synchronization error is measured between the Kinect depth camera and a hardware-synchronized color camera. Dataset **B**, from [56], contains synthetic depthmaps which are synchronous with their respective color images. An unsynchronized depth camera in dataset **B** is simulated by offsetting the depthmap sequence from the color image sequence.

For our method, Eq. (9) is calculated for each moving correspondence point, and the mean of these offsets is taken as the estimated Δt. Point coordinates in dataset **A** images are obtained by the Matlab implementation of checkerboard detector by Geiger et al. [57], and by SURF [58] feature detector in dataset **B**. The synchronization method by Albl et al. [45] was chosen because it can handle subframe accuracy of Δt_n estimation, is shown to perform well on synthetic data, compares favourably against other state-of-art methods, has a publicly available reference implementation, and allows for manually identified correspondence points as well as auto-detected generic image features. Since it relies on motion tracks in image sequences, for each Δt_n estimation we provide the frames in range $[n-10, n+10]$. This satisfies the requirement mentioned in [45], specifying that the motion track length in each sequence has to be longer than the synchronization offset.

IV-C Synchronization Error Compensation Test

The proposed synchronization error compensation method is compared with sparse-correspondence based geometric image morphing [50], two dense optical flow based interpolation methods [53], [59], and one displacement field interpolation method based on Thirion's demons algorithm [60]. These methods were selected due to their similarity with the proposed method and their inclusion in publicly available image processing tools (such as Matlab and OpenCV). The sparse-correspondence morphing

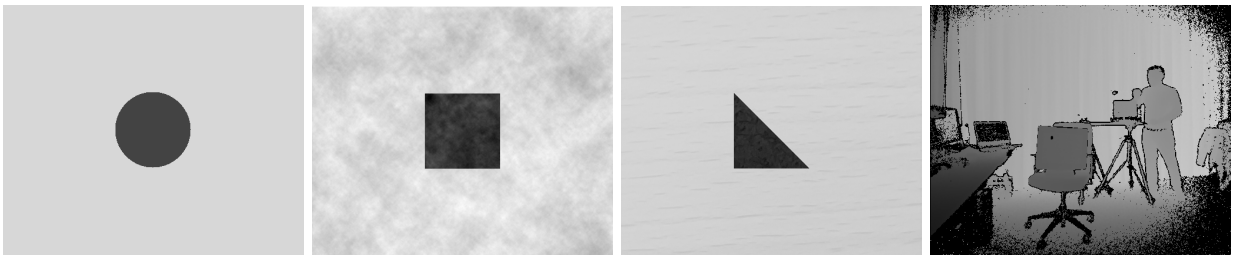


Fig. 5. Artificially generated depthmap proxies and a depthmap from Kinect V2 depth camera. Depthmap proxies vary foreground object shape and texture. Kinect depthmap has noise in view periphery and on object edges. Nearer pixels have a lower depth value.

method relies on correspondences from Scale-Invariant Feature Transform (SIFT) [61] with RANSAC [47] correspondence filtering.

All methods are evaluated on real and artificial depthmap sequences, shown in Fig. 5. The artificial depthmaps vary between simple shapes and textures. A set of depthmaps is generated with no foreground or background texture, to demonstrate a worst-case scenario for optical flow estimation. Another set of depthmaps uses a Perlin-noise texture for foreground and background, and another uses natural wood and leather texture. The MVD sequence provides a source of natural noise and motion. The objective evaluation metric is the Mean Squared Error (MSE) between the ground truth depthmap and the compensated depthmap. On artificially generated depthmaps, δ is checked in interval $[0.1, 0.9]$. On real depthmaps, $\delta = 0.5$ in order to ensure that we have a ground truth depthmap whilst retaining a short time interval between t_n and t_{n+1}. We also provide a subjective comparison between the proposed method and the non-compensated case, to provide context for the objective results. This is done by using the DIBR [55] algorithm and depth-based foreground isolation on the MVD sequences mentioned in Section IV-B.

V Results and Analysis

Three sets of experiments have been described in section IV. The first experiment investigates the link between depth uncertainty and reprojected image quality. The second experiment investigates the accuracy of the proposed synchronization error estimation. The third experiment investigates the proposed synchronization error compensation method, and the difference between using unsynchronized and re-synchronized depthmaps.

V-A Depth Uncertainty

Figure 6 shows the mapping between a system's depth uncertainty, based on varying synchronization offsets. In a camera system with parallel-oriented cameras (dataset B), the depth uncertainty is overall larger than in a system with converged cameras (dataset A), reinforcing the conclusion from [18]. Figure

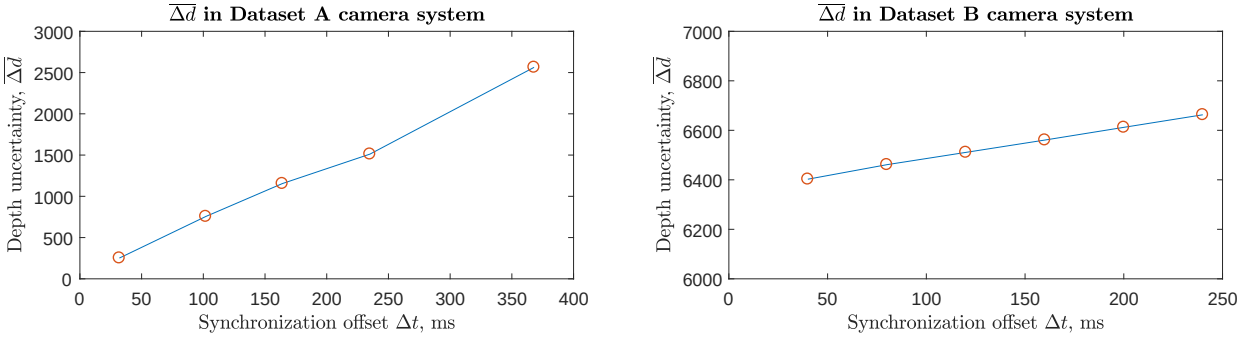


Fig. 6. Depth uncertainty $\overline{\Delta d}$ dependence on varying Δt, for camera systems used to record datasets A and B.

7 shows how synchronization offset affects the objective reprojected image quality. In dataset B, the results confirm the expected behavior - an increase in synchronization offset causes a relative decrease in quality, on the order of 2 to 3 dB per 100 ms. The change in additional parameters (presence of interpolation-based disocclusion inpainting, baseline between original camera and reprojection target, temporal ordering of image/depthmap) cause a global quality change, but do not disrupt the additional influence from synchronization offset. The presence or absence (”F” or ”NF” in Fig. 7) of inpainting has a notably larger influence on Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index (SSIM) and MSE than synchronization offset. Increasing the distance between reprojection source and destination cameras (”2” and ”3”) decreases overall quality by < 1 dB, whereas the sign of the temporal offset between color image and depthmap has no significant effect. Contrary to dataset B, the results for dataset A in Fig. 7 show no relation to changes in the synchronization offset, and indicate very low overall reprojection quality.

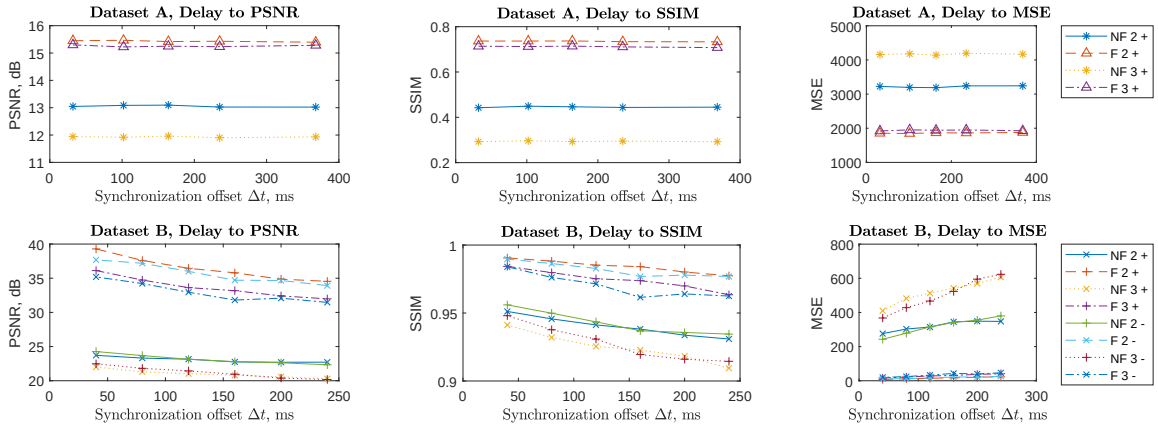


Fig. 7. DIBR-based view render quality, given Δt synchronization offset between depthmap and texture. ”NF”: DIBR with no disocclusion filling. ”F”: DIBR with disocclusion filling. ”2”,”3”: images on camera 1 projected, respectively, to image plane of cameras 2, 3. ”+”,”-”: unsynchronized depthmap for DIBR chosen x ms before/after the color image.

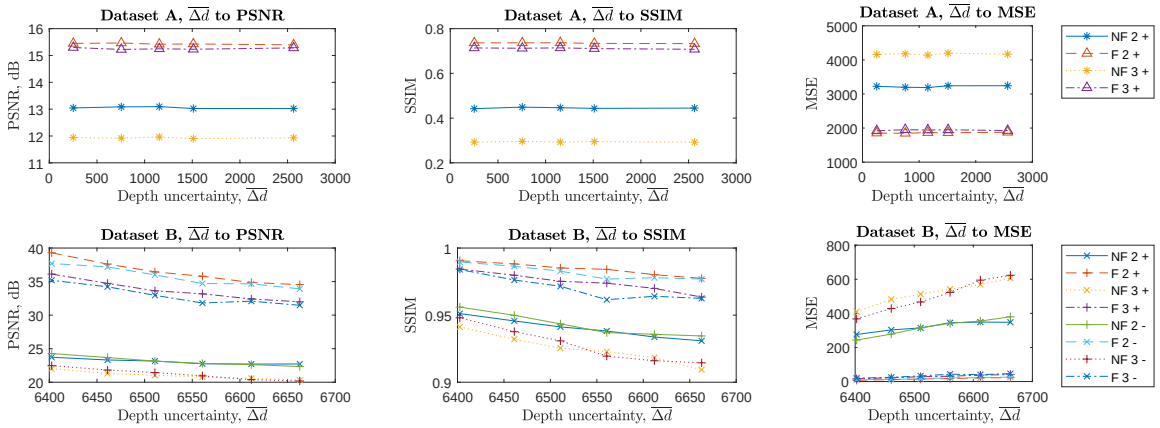


Fig. 8. Relation between camera system depth uncertainty $\overline{\Delta d}$ and reprojected image quality. "NF": DIBR with no disocclusion filling. "F": DIBR with disocclusion filling. "1","2": images on camera 1 projected, respectively, to image plane of cameras 2, 3. "+","-": unsynchronized depthmap for DIBR chosen x ms before/after the color image.

Since Δt to $\overline{\Delta d}$ is a linear mapping in Fig. 6, then Fig. 8 shows the same result behaviour as Fig. 7. In case of dataset B, the results show that increasing the system's overall depth uncertainty causes a measurable drop in objective reprojection quality, for a given starting level of reprojection quality. However, dataset A shows a different situation, indicating that a previously untreated parameter can disrupt the relation between $\overline{\Delta d}$ and objective reprojection quality. Figure 9 shows an example of reprojected image from both datasets, compared against the reference image recorded by the target camera. Mismatches between reference pixel value and reprojected pixel value are highlighted in green and purple. The reprojection sample of dataset A has significant alignment errors even on non-moving segments, such as the scene background and time-control monitor. The sample shown in Fig. 9 is representative of all frames in dataset A. Such errors in DIBR are caused by incorrect camera calibration matrices, and, according to Fig. 7, are sufficient to disrupt any pixel-based quality metric.

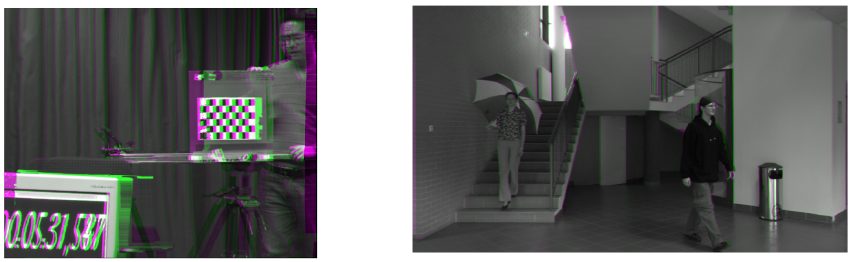


Fig. 9. DIBR reprojection sample for Datasets A (left) and B (right), using the datasets' reported calibration matrices and low depthmap synchronization error (32 ms in Dataset A, 40 ms in Dataset B). Green & magenta areas indicate mismatch in pixel values between the reprojected image and the true destination image.

V-B Synchronization Error Estimation

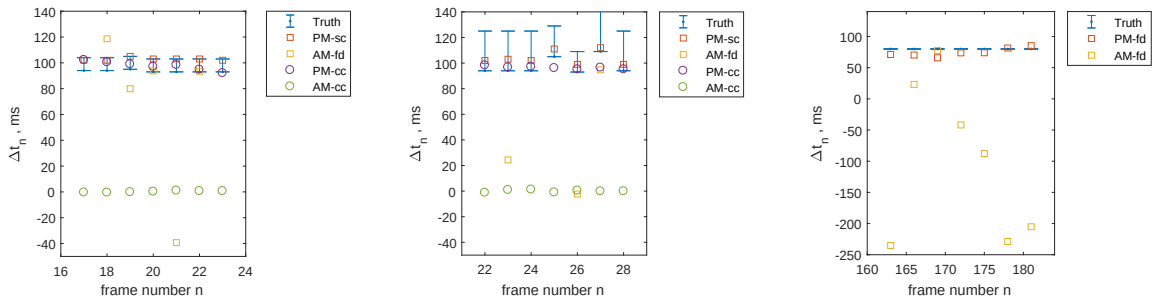


Fig. 10. Estimated and true synchronization delay Δt_n between an RGB camera and a depth camera at frame n. Left: Dataset A, sequence 1, both cameras have constant framerate at 7.5 Hz. Ground truth known with 10 ms accuracy (due to in-scene clock refresh rate). Middle: Dataset A, sequence 2, depth camera has variable framerate. RGB camera has longer exposure time, so ground truth Δt_n accuracy is reduced. Right: Dataset B, $\Delta t = 80$ ms simulated by frame offset between RGB images and depthmaps. "Truth": ground truth min and max for Δt_n. "PM": proposed method. "AM": method by Albl et al. [45]. "fd": moving points located by feature detection algorithms. "cc" & "sc": moving points located by checkerboard-corner detection algorithms, respectively for all checkerboard points and single manually-selected point.

Figure 10 shows that the proposed method manages to estimate synchronization offsets within the ground truth limits for the case of dataset A, and near the ground truth of dataset B. In Fig. 10, left, the average difference between estimated Δt_n and ground truth Δt_n is 9.3 ms to the lower ground truth bound, and 0.7ms to the upper ground truth bound. Considering the framerate in the test scenario (7.5 Hz), the proposed method manages to compensate 92.9% of the synchronization error.

In the middle scenario (Fig. 10), ground truth intervals are larger due to longer RGB camera exposure times, and due to varying depth camera framerate (caused by enabling bilateral noise filter in the kinect SDK during recording). Estimation error is 6.5 ms to the lower bound and 22.0 ms to the upper bound, on average. This corresponds to a compensation of 95.2% and 83%, respectively, relative to the RGB camera framerate. For all frames, estimated Δt_n lies between the lower and upper bounds of Δt_n ground truth.

In the case of dataset B, the average estimation error is approx. 9.6 ms to the ground truth. In all cases, the error of the proposed method's synchronization offset estimation is less than the exposure time of one frame of the respective RGB cameras, in both dataset A and B.

The compared method [45] shows nearly random behaviour when using the bundled feature-detector ("AM-fd" in Fig. 10), likely due to errors in feature matching. In order to eliminate feature mismatch errors, feature points are taken from checkerboard corner detection on dataset A sequences. In the checkerboard-corner detection case, the compared method estimates $\Delta t = 0$ (Fig. 10 left, middle). This implies that the compared method does not manage to handle smooth, non-chaotic movement in scenes, which has been an unexplored scenario in [45]. In contrast, the proposed method does not require temporally long or chaotic motion, and therefore can give more accurate estimates for Δt.

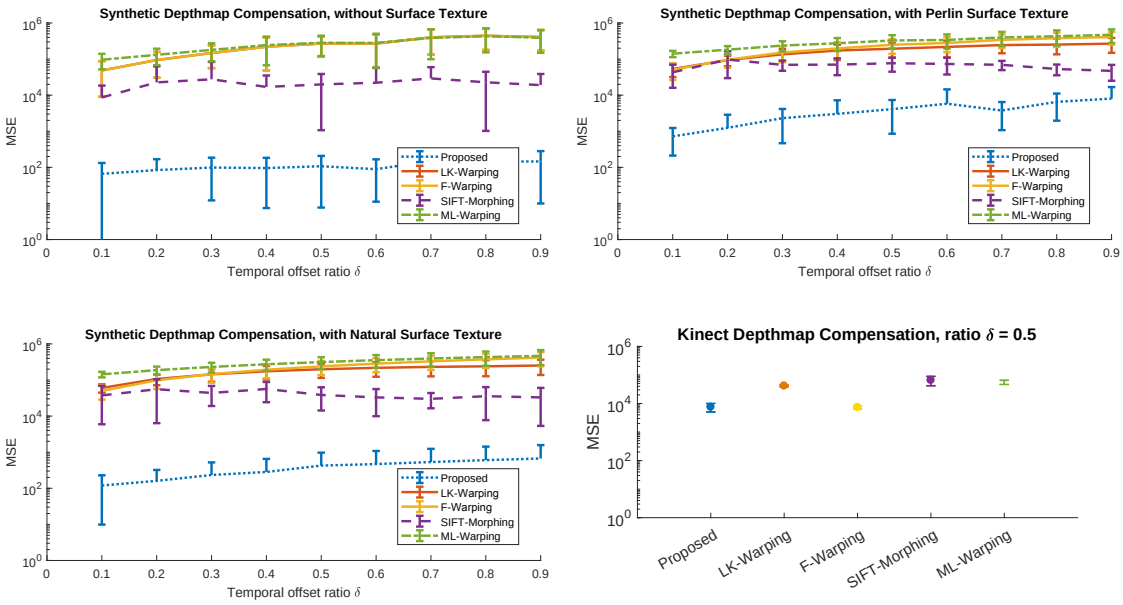


Fig. 11. Synchronization error compensation for 5 methods, conducted on three sets of artificially generated (synthetic) depthmap sequences, and one set of sequences from a Kinect depth camera. Synchronization error between depthmaps is set by the ratio δ. The Proposed method is compared to optical flow based interpolation (LK-Warping [59], F-Warping [53]), Image morphing [50] based on SIFT [61], and Displacement field based interpolation (ML-Warping) [60]. Lines show mean MSE, whiskers show the standard deviation of MSE from multiple iterations with different input depthmaps.

V-C Synchronization Error Compensation

Figure 11 shows the objective evaluation results for the five compared methods that can compensate synchronization error by reconstructing the target image. The synthetic depthmaps are problematic for methods that rely on optical flow, due to low overall scene contrast. The combination of SIFT, RANSAC and image morphing is able to ensure a certain level of quality independent of δ, however it does not outperform the proposed method. In case of real depthmaps, the performance is closer for all methods. The increased MSE can partly be attributed to the presence of noise in the recorded Kinect depthmaps. The more complex scene and significantly smaller movement from frame to frame allows optical-flow based methods to do a better synchronization error compensation, on par with the proposed method.

Figure 12 shows the difference between using unsynchronized depth ($\Delta t = 102ms$) and compensated depth for foreground extraction and reprojection. The foreground isolation example shows that the compensated depth provides a better alignment of depth and color data for moving objects. The reprojection example with compensated depth shows that disocclusion cracks are aligned with object boundaries (checkerboard, hand, face edges). Using non-compensated depth causes disocclusion cracks to appear over the moving foreground objects. No synchronized reference depth is available for this dataset, however comparing top and bottom row of Fig. 12 shows that the synchronization error compensation process has not noticeably degraded the depthmap resolution.

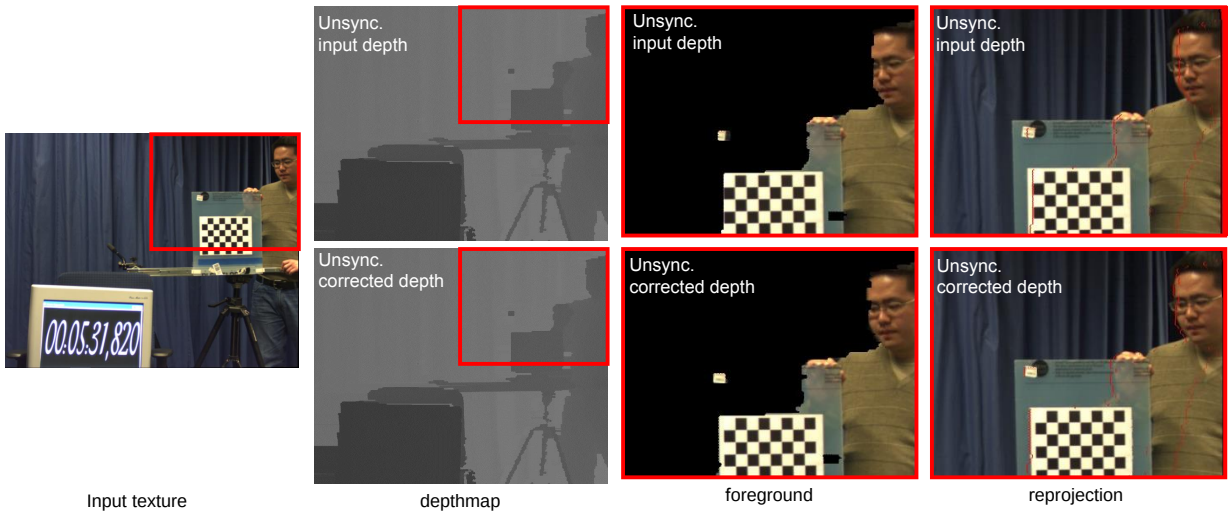


Fig. 12. Dataset A. Depth-based foreground isolation and view reprojection, using depth with synchronization error $\Delta t = 102$ ms (top row) and same depth re-synchronized by the proposed compensation method (bottom row). Dataset's depthmap captured by ToF camera and warped to input texture's respective camera view.

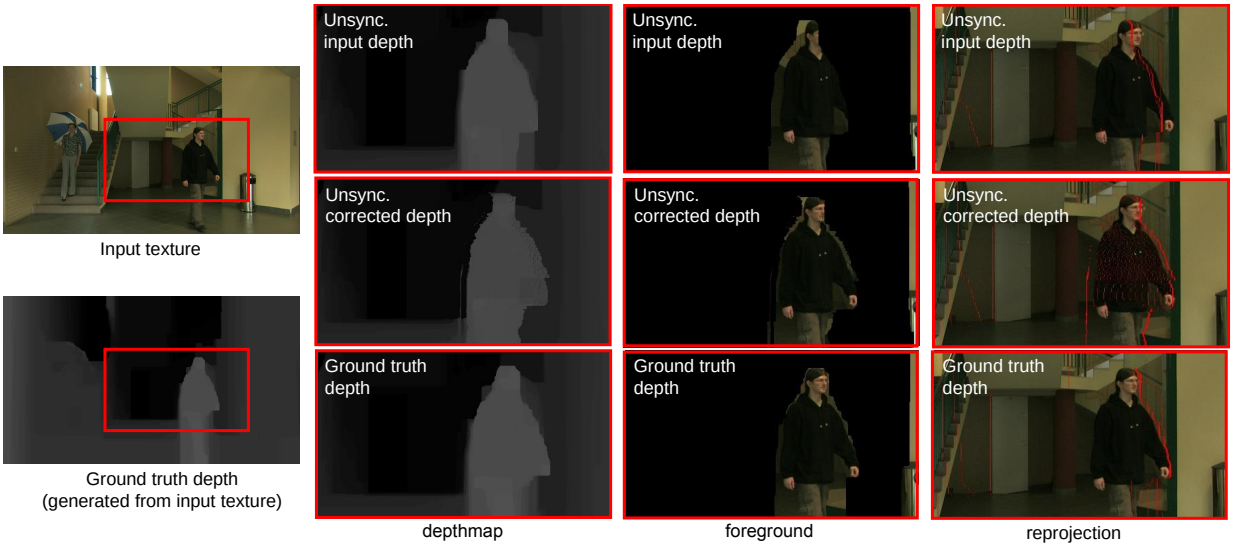


Fig. 13. Dataset B. Depth-based foreground isolation and view reprojection, using depth with synchronization error $\Delta t = 80$ ms (top row), same unsynchronized depth after compensation by the proposed method (middle row), and ground truth depth from the dataset (bottom row). Dataset's depthmaps generated synthetically from input textures [56].

Figure 13 shows the same setup with dataset B, for which a ground truth depthmap is also available. As seen in the foreground extraction column, the unsynchronized input depth results in noticeable misalignment error between depth and texture. Using the compensated depth data improves the alignment between depth and color data. The reprojection example again shows that using unsynchronized depth causes disocclusion cracks on the foreground object, whereas the compensated and ground truth

depthmaps produce disocclusions on moving object boundaries.

VI Conclusion

In this work, we addressed two research questions: first, whether the impact of synchronization errors in a multi-camera system can be determined ahead-of-time; second, whether the synchronization errors can e estimated and compensated in a way that benefits conventional depth-based rendering tasks. In response to these questions, we described the depth uncertainty model as a way to parametrize the impact of synchronization error. Next, we proposed a method for finding the sub-frame synchronization error in multicamera systems with depth and color cameras, and a method for compensating the synchronization error in depthmap sequences. The proposed synchronization error estimation method relies on movement of identifiable objects in scene, and works for sub-frame synchronization errors. The proposed depth compensation method relies on segmentation and guided interpolation of depthmaps for the desired time instant.

Experiments of depth uncertainty modeling, synchronization offset estimation, and synchronization error compensation have been performed, using two real-world datasets with controlled and human motion, different camera configurations, layouts, and synchronization errors. The synchronization error compensation has also been evaluated on artificially generated depthmaps.

Through depth uncertainty model simulations and DIBR reprojections, we demonstrated that depth uncertainty can imply degradation in image quality, when integrating unsynchronized depth and texture data. The connection holds as long as the camera system's calibration is sufficiently accurate. Through synchronization error estimation tests, we showed that the proposed method outperforms a state-of-the-art sequence alignment approach. Whereas the compared method is based on long feature trajectories through time, our method does not require chaotic dense motion or large temporal video buffer. Through synchronization error compensation tests, we validated the proposed compensation process, and showed the difference between using synchronized, unsynchronized, and compensated depth data in DIBR and depth-based selection processes.

Acknowledgement

This work has received funding from grant 6006-214-290174 from Rådet för Utbildning på Forskarnivå (FUR), Mid Sweden University, from LIFE project grant 20140200 of the Knowledge Foundation, Sweden, and from the European Union's Framework Programme for Research and Innovation Horizon 2020 (2014-2020) under the Marie Skłodowska-Curie Actions Grant Agreement No. 676401. The authors would like to thank Cenek Albl for sharing the code for [45].

References

[1] H. Joo, H. Liu, L. Tan, L. Gui, B. Nabbe, I. Matthews, T. Kanade, S. Nobuhara, and Y. Sheikh, "Panoptic studio: A massively multiview system for social motion capture," in *IEEE International Conference on Computer Vision (ICCV)*, 2015.

[2] K. Berger, K. Ruhl, Y. Schroeder, C. Bruemmer, A. Scholz, and M. A. Magnor, "Markerless motion capture using multiple color-depth sensors." in *VMV*, 2011, pp. 317–324.

[3] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox, "Rgb-d mapping: Using kinect-style depth cameras for dense 3d modeling of indoor environments," *The International Journal of Robotics Research*, vol. 31, no. 5, pp. 647–663, 2012.

[4] C. Kerl, J. Stuckler, and D. Cremers, "Dense continuous-time tracking and mapping with rolling shutter rgb-d cameras," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 2264–2272.

[5] L. Heng, G. H. Lee, and M. Pollefeys, "Self-calibration and visual slam with a multi-camera system on a micro aerial vehicle," *Autonomous Robots*, vol. 39, no. 3, pp. 259–277, 2015.

[6] J. Lopez-Moreno, J. Jimenez, S. Hadap, K. Anjyo, E. Reinhard, and D. Gutierrez, "Non-photorealistic, depth-based image editing," *Computers & Graphics*, vol. 35, no. 1, pp. 99–111, 2011.

[7] M. Ziegler, A. Engelhardt, S. Müller, J. Keinert, F. Zilly, S. Foessel, and K. Schmid, "Multi-camera system for depth based visual effects and compositing," in *ACM European Conference on Visual Media Production (CVMP)*, 2015, p. 3.

[8] M. Domański, A. Dziembowski, D. Mieloch, A. Łuczak, O. Stankiewicz, and K. Wegner, "A practical approach to acquisition and processing of free viewpoint video," in *Picture Coding Symposium (PCS), 2015*. IEEE, 2015, pp. 10–14.

[9] C. Zhu, Y. Zhao, L. Yu, and M. Tanimoto, *3D-TV System with depth-image-based rendering*. Springer, 2014.

[10] S.-H. Ou, C.-H. Lee, V. S. Somayazulu, Y.-K. Chen, and S.-Y. Chien, "On-line multi-view video summarization for wireless video sensor network," *IEEE Journal of Selected Topics in Signal Processing*, vol. 9, no. 1, pp. 165–179, 2015.

[11] R. Du, S. Bista, and A. Varshney, "Video fields: fusing multiple surveillance videos into a dynamic virtual environment," in *Proceedings of the 21st International Conference on Web3D Technology*. ACM, 2016, pp. 165–172.

[12] K. Berger, S. Meister, R. Nair, and D. Kondermann, "A state of the art report on kinect sensor setups in computer vision," in *Time-of-Flight and Depth Imaging. Sensors, Algorithms, and Applications*. Springer, 2013, pp. 257–272.

[13] V. Gandhi, J. Čech, and R. Horaud, "High-resolution depth maps based on TOF-stereo fusion," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2012, pp. 4742–4749.

[14] M. Munaro, F. Basso, and E. Menegatti, "Openptrack: Open source multi-camera calibration and people tracking for rgb-d camera networks," *Robotics and Autonomous Systems (RAS)*, vol. 75, pp. 525–538, 2016.

[15] F. Fleuret, J. Berclaz, R. Lengagne, and P. Fua, "Multicamera people tracking with a probabilistic occupancy map," *IEEE transactions on pattern analysis and machine intelligence*, vol. 30, no. 2, pp. 267–282, 2008.

[16] A. Elhayek, C. Stoll, N. Hasler, K. I. Kim, H.-P. Seidel, and C. Theobalt, "Spatio-temporal motion tracking with unsynchronized cameras," in *2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2012, pp. 1870–1877.

[17] H. Sarbolandi, D. Lefloch, and A. Kolb, "Kinect range sensing: Structured-light versus time-of-flight kinect," *Computer Vision and Image Understanding (CVIU)*, vol. 139, pp. 1–20, 2015.

[18] E. Dima, M. Sj"oström, and R. Olsson, "Modeling depth uncertainty of desynchronized multi-camera systems," in *2017 International Conference on 3D Immersion (IC3D)*. IEEE, December 2017.

[19] R. Nair, K. Ruhl, F. Lenzen, S. Meister, H. Schäfer, C. S. Garbe, M. Eisemann, M. Magnor, and D. Kondermann, "A survey on time-of-flight stereo fusion," in *Time-of-Flight and Depth Imaging. Sensors, Algorithms, and Applications*. Springer, 2013, pp. 105–127.

[20] P. Zanuttigh, G. Marin, C. Dal Mutto, F. Dominio, L. Minto, and G. M. Cortelazzo, "Data fusion from depth and standard cameras," in *Time-of-Flight and Structured Light Depth Cameras*. Springer, 2016, pp. 161–196.

[21] A. Noda, Y. Yamakawa, and M. Ishikawa, "Frame synchronization for networked high-speed vision systems," in *SENSORS*. IEEE, 2014, pp. 269–272.

[22] C. Albl, Z. Kukelova, A. Fitzgibbon, J. Heller, M. Smid, and T. Pajdla, "On the two-view geometry of unsynchronized cameras," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, July 2017.

[23] G. Litos, X. Zabulis, and G. Triantafyllidis, "Synchronous image acquisition based on network synchronization," in *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2006, pp. 167–167.

[24] M. A. Mughal and O. Juhlin, "Context-dependent software solutions to handle video synchronization and delay in collaborative live mobile video production," *Personal and ubiquitous computing*, vol. 18, no. 3, pp. 709–721, 2014.

[25] Y. Yoon, M. Kim, B. Lee, and K. Go, "Temporal synchronization scheme in live 3d video streaming over ieee 802.11 wireless networks," in *IEEE 15th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)*. IEEE, 2014, pp. 1–7.

[26] R. Latimer, J. Holloway, A. Veeraraghavan, and A. Sabharwal, "Socialsync: Sub-frame synchronization in a smartphone camera network," in *European Conference on Computer Vision*. Springer, 2014, pp. 561–575.

[27] H. Haberdar and S. K. Shah, "Video synchronization as one-class learning," in *Proceedings of the 27th Conference on Image and Vision Computing New Zealand*. ACM, 2012, pp. 469–474.

[28] P. Shrstha, M. Barbieri, and H. Weda, "Synchronization of multi-camera video recordings based on audio," in *Proceedings of the 15th ACM international conference on Multimedia*. ACM, 2007, pp. 545–548.

[29] F. Schweiger, G. Schroth, M. Eichhorn, A. Al-Nuaimi, B. Cizmeci, M. Fahrmair, and E. Steinbach, "Fully automatic and frame-accurate video synchronization using bitrate sequences," *IEEE Transactions on Multimedia (TMM)*, vol. 15, no. 1, pp. 1–14, 2013.

[30] F. Diego, D. Ponsa, J. Serrat, and A. M. López, "Video alignment for change detection," *IEEE Transactions on Image Processing*, vol. 20, no. 7, pp. 1858–1869, 2011.

[31] Y. Caspi and M. Irani, "Spatio-temporal alignment of sequences," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 11, pp. 1409–1424, 2002.

[32] P. Shrestha, H. Weda, M. Barbieri, and D. Sekulovski, "Synchronization of multiple video recordings based on still camera flashes," in *Proceedings of the 14th ACM international conference on Multimedia*. ACM, 2006, pp. 137–140.

[33] C. Lei and Y.-H. Yang, "Tri-focal tensor-based multiple video synchronization with subframe optimization," *IEEE Transactions on Image Processing*, vol. 15, no. 9, pp. 2473–2480, 2006.

[34] T. Tuytelaars and L. Van Gool, "Synchronizing video sequences," in *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2004)*, vol. 1. IEEE, 2004, pp. I–I.

[35] C. Lu and M. Mandal, "A robust technique for motion-based video sequences temporal alignment," *IEEE Transactions on Multimedia*, vol. 15, no. 1, pp. 70–82, 2013.

[36] G. D. Evangelidis and C. Bauckhage, "Efficient subframe video alignment using short descriptors," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 10, pp. 2371–2386, 2013.

[37] D. Pundik and Y. Moses, "Video synchronization using temporal signals from epipolar lines," in *European Conference on Computer Vision*. Springer, 2010, pp. 15–28.

[38] C. Dai, Y. Zheng, and X. Li, "Subframe video synchronization via 3d phase correlation," in *IEEE International Conference on Image Processing (ICIP)*. IEEE, 2006, pp. 501–504.

[39] F. Padua, R. Carceroni, G. Santos, and K. Kutulakos, "Linear sequence-to-sequence alignment," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 2, pp. 304–320, 2010.

[40] K. Ruhl, F. Klose, C. Lipski, and M. Magnor, "Integrating approximate depth data into dense image correspondence estimation," in *Proceedings of the 9th European Conference on Visual Media Production*. ACM, 2012, pp. 26–31.

[41] M. Noguchi and T. Kato, "Geometric and timing calibration for unsynchronized cameras using trajectories of a moving marker," in *IEEE Workshop on Applications of Computer Vision (WACV'07)*. IEEE, 2007, pp. 20–20.

[42] M. Nischt and R. Swaminathan, "Self-calibration of asynchronized camera networks," in *IEEE 12th International Conference on Computer Vision Workshops (ICCV Workshops)*. IEEE, 2009, pp. 2164–2171.

[43] K. Ruhl, B. Hell, F. Klose, C. Lipski, S. Petersen, and M. Magnor, "Improving dense image correspondence estimation with interactive user guidance," in *Proceedings of the 20th ACM International Conference on Multimedia*. ACM, Oct 2012, pp. 1129–1132.

[44] F. Klose, C. Lipski, K. Ruhl, B. Meyer, and M. Magnor, "A toolchain for capturing and rendering stereo and multi-view datasets," in *Proceedings of the International Conference on 3D Imaging (IC3D)*, Dec 2011, pp. 1–7.

[45] C. Albl, Z. Kukelova, A. Fitzgibbon, J. Heller, M. Smid, and T. Pajdla, "On the two-view geometry of unsynchronized cameras," *arXiv preprint arXiv:1704.06843*, 2017.

[46] Z. Kukelova, M. Bujnak, and T. Pajdla, "Automatic generator of minimal problem solvers," *Computer Vision–ECCV 2008*, pp. 302–315, 2008.

[47] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," in *Readings in computer vision*. Elsevier, 1987, pp. 726–740.

[48] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge University Press, 2003.

[49] V. J. Lumelsky, "On fast computation of distance between line segments," *Information Processing Letters*, vol. 21, no. 2, pp. 55–61, 1985.

[50] S. M. Seitz and C. R. Dyer, "View morphing," in *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*. ACM, 1996, pp. 21–30.

[51] G. Wolberg, "Image morphing: a survey," *The Visual Computer*, vol. 14, no. 8, pp. 360–372, 1998.

[52] T. Beier and S. Neely, "Feature-based image metamorphosis," in *ACM SIGGRAPH Computer Graphics*, vol. 26, no. 2. ACM, 1992, pp. 35–42.

[53] G. Farnebäck, "Two-frame motion estimation based on polynomial expansion," *Image Analysis*, pp. 363–370, 2003.

[54] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert, "High accuracy optical flow estimation based on a theory for warping," *Computer Vision-ECCV 2004*, pp. 25–36, 2004.

[55] C. Fehn, "Depth-image-based rendering (dibr), compression, and transmission for a new approach on 3d-tv," in *Electronic Imaging 2004*. International Society for Optics and Photonics, 2004, pp. 93–104.

[56] M. Domañski, T. Grajek, K. Klimaszewski, M. Kurc, O. Stankiewicz, J. Stankowski, and K. Wegner, "Poznan multiview video test sequences and camera parameters," *ISO/IEC JTC1/SC29/WG11 MPEG*, p. M17050, 2009.

[57] A. Geiger, F. Moosmann, Ö. Car, and B. Schuster, "Automatic camera and range sensor calibration using a single shot," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2012, pp. 3936–3943.

[58] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (surf)," *Computer vision and image understanding*, vol. 110, no. 3, pp. 346–359, 2008.

[59] T. Kanade, P. Rander, and P. Narayanan, "Virtualized reality: Constructing virtual worlds from real scenes," *IEEE MultiMedia*, vol. 4, no. 1, pp. 34–47, 1997.

[60] J.-P. Thirion, "Image matching as a diffusion process: an analogy with maxwell's demons," *Medical image analysis*, vol. 2, no. 3, pp. 243–260, 1998.

[61] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.